Lecture 28

Integrated Cognitive Architectures: A Survey
Abstract

- A review of six cognitive architectures
  - SOAR, ACT-R, ICARUS, BDI, the subsumption architecture and CLARION
  - Functional comparison through cognitive components, including perception, memory, goal representation, planning, problem solving, reasoning, learning, and relevance to neurobiology
  - The range of benchmarks and applications that these architectures have been applied to

- No single cognitive architecture has provided a full solution with the level of human intelligence, but important design principles have emerged, pointing to promising directions towards generic and scalable architectures with close analogy to human brains.
Integrated cognitive architecture (ICA)

- A single system that is capable of producing all aspects of behavior, while remaining constant across various domains and knowledge bases (Newell 1990; Anderson et al. 2004)
- Consisting of many modules (or components) working together to produce a behavior, which are
  - representations of knowledge,
  - memories for storage of content and
  - processes utilizing and acquiring knowledge.
- To explain a wide range of human behavior, and to mimic the broad capabilities of human intelligence
- Research on ICA:
  Interdisciplinary by nature, spanning the fields of AI, cognitive psychology, and neurobiology
An overview of the six cognitive architectures
Introduction

- **SOAR (State, operator, and result)** (Laird et al. 1986)
  - One of the earliest and most extensively developed AI architectures in the history

- **ACT-R (Anderson et al.’04) and ICARUS** (Langley and Choi ’06)
  ACT-R: Adaptive control of thought-rational
  - Cognitive systems developed with the primary aim of producing AI mimicking human cognition
  - Different from SOAR by their strong emphasis of producing a psychologically motivated cognitive model, though they share many features of classical AI, including symbolic representation, production rule based inference, and means-end analysis for problem solving
Belief-Desire-Intention architecture (Bratman et al. 1988)
- A framework, incorporating beliefs, desires and intentions, for designing intelligent autonomous agents
- A special focus on intentions, representing an agent’s commitments to carry out certain plans of actions

Subsumption architecture (Brooks 1999)
- Behavior based and thus does not contain any problem solving or learning module
- The idea of higher layers subsuming lower layers has its root from neurobiology

CLARION (Sun and Peterson 1996)
CLARION: Connectionist Learning with Adaptive Rule Induction ON-line
- A hybrid model integrating both symbolic and connectionist information processing
- Based on NNs as well as cognitive psychology
1. SOAR

- To handle the full range of capabilities of an intelligent agent through a general mechanism of learning from experience.
- Used for the understanding of the mechanisms required for intelligent behavior and the incorporation of the mechanisms to form a general cognitive architecture in classical AI.
1. SOAR

1.1 SOAR Architecture

- Consisting of the various memory structures including LTM (long term memory) and WM (working memory), and a decision making mechanism linking perception to action.

- Information from the environment is made available in the WM via perception, allowing appropriate actions to be chosen during domain-independent problem solving.

- Procedural memory provides the knowledge of performing tasks.

- Semantic memory stores general facts about the world and is considered as declarative knowledge.

- Episodic memory contains specific memory of an event experienced.
1. SOAR

- WM houses all the knowledge that are relevant to the current situation.
  - It contains the goals, perceptions, hierarchy of states, and operators.

- The states (and sub-states) give the information on the current situation.

- The operator provides the steps to apply during problem solving, while the goal directs the architecture into the desired state.

- The content of the WM (WM elements) can trigger both the retrieval of relevant knowledge from the LTM into the WM, and motor actions.
1.2 Functions and Processes

- Soar supports a number of problem solving methods.
  - Through the means-ends analysis, the system selects and applies operators to result in a new state that is closer to the desired state.
  - To bring the system closer to its goal, Soar implements a *five-phase decision cycle*, constituting of input, elaboration, decision, application, and output.
  - The main function of the decision cycle is to choose the next operator to apply.
Elaboration phase (EP)
- Percepts are added into the WM in the input phase for use.
- Production rules are then matched with the WM elements to bring knowledge relevant to the current problem into the WM.
- Meanwhile, preferences, which encode the procedural search-control knowledge, are created to act as recommendations for selection of appropriate operators.
- The EP continues till the firing of rules in the LTM ceases, ensuring that all knowledge relevant to the current situation is considered before a decision is made.

When the EP reaches a quiescence, the decision cycle proceeds to the decision phase, wherein the preferences are evaluated.

The better of the suggested operators are chosen and applied during the application phase.

A motor action is then performed during the output phase as a result of applying the selected operator.
An impasse is encountered whenever the procedural knowledge is inadequate for problem solving.

- Four types of impasse: no-change, tie, conflict, and rejection.
  - The *no-change impasse* occurs when the EP enters a quiescence w/o any suggestion.
  - The *tie impasse* refers to the situation whereby no object is superior over the others.
  - The *conflict impasse* occurs for cases in which two or more candidate objects are better than each other.
  - The *rejection impasse* is encountered when all operators are rejected.

Soar learns from any impasse encountered.

- The learning mechanisms include chunking, reinforcement learning, episodic memory, and semantic memory.
- A successful resolution of an impasse results in the termination of a goal or subgoal, which in turn leads to formation of chunks.
1. SOAR

- The chunking mechanism to add new production rules into the LTM.
  - These chunks are used whenever a similar situation is encountered, thereby avoiding the same impasse and improving the performance of the agent in the future.
  - Soar also receives rewards from successes and punishments from failures, allowing the agent to undergo reinforcement learning.
  - Any operator resulting in a reward upon an execution is given a positive reinforcement, and such operators are more likely to be selected in the future.

- The episodic and semantic memories store information on the agent’s past experiences, and therefore both are used as additional cues to select applicable operators.
2. ACT-R

- To develop a human cognition model, using empirical data derived from experiments in cognitive psychology and brain imaging
- To provide a step by step simulation of human behavior for detailed understanding of human cognition
2. ACT-R

2.1 ACT-R Architecture

- Information from the external environment and knowledge stored in the memories work conjunctively to select actions for execution to satisfy the goal(s) of the agent.

- *Four basic modules*: visual, manual, declarative memory, and goal modules
  - The *visual module* is needed for identifying objects in the visual field.
  - The *manual module* is used for the control of the hand actuators.
  - Information from memories can be retrieved via the *declarative memory module*.
  - The *goal module* keeps track of the agent’s current goals, and enables the maintenance of the agent’s thought in the absence of supporting external stimuli.

- All the above modules are coordinated through the central production system, which is identified to be the basal ganglia in human brain.
Production rules are implemented in the central production system, corresponding to striatum, pallidum and thalamus, as well as the associated connections with the various buffers.

Four buffers: goal, retrieval, manual, and visual buffers

- The goal buffer helps to keep track of one’s internal state during problem solving and it has been identified that the dorsolateral prefrontal cortex (DLPFC) holds the role of a goal buffer.
- The declarative memory retrieved from the LTM store is passed to the retrieval buffer, which is associated with the ventrolateral prefrontal cortex (VLPFC).
- The manual buffer for controlling the agent’s hand is associated with the motor and somatosensory cortical areas, which are the two areas in the brain that control and monitor hand movement.
- The visual buffers in this model include both the dorsal ‘where’ path of visual system and the ventral ‘what’ system.
2.2 Functions and Processes

- This model assumes a mixture of parallel and serial processing.
  - Parallelism occurs within each module as well as between different modules.
- There are two levels of serial bottleneck in this model.
  - The first bottleneck is that the content of any buffer is only limited to a single declarative unit of knowledge, known as chunks in this model.
  - The second is that only a single production is fired in every cycle.
- The knowledge can be stored in either procedural memory or declarative memory.
  - Knowledge in declarative memory is represented as chunks, while knowledge in procedural memory is available in terms of production rules.
  - A chunk is only retrieved when its activation level raises above a threshold level.
  - Whenever a production rule matches the chunk retrieved, the production rule is fired.
3. ICARUS

- Its root in designing physical and embodied agents, through integrating perception and action with cognition
- Aims to unify reactive execution with problem solving, combine symbolic structures with numeric utilities, and learn structures and utilities in a cumulative manner
3. ICARUS

3.1 ICARUS Architecture

- The perceptual buffer is involved in the temporary storage of percepts.
- Conceptual memory can be further classified into STCM (short term conceptual memory) and LTCM (long term conceptual memory).
  - The STCM contains the set of active inferences describing the relations among the objects perceived.
  - The LTCM consists of the known conceptual structures describing objects or classes of the environmental situations.
- The skill memory is subdivided into STSM (short term skill memory) and LTSM (long term skill memory).
  - All the skills that can be executed by the ICARUS agent are stored in the LTSM.
  - The chosen skill to be implemented is brought into the STSM.
3.2 Functions and Processes

- Conceptual inference is the mechanism responsible for matching the conceptual structures against the percepts and beliefs.
  - It is dependent on the content and representation of elements that have been stored in both the short term and long term memories.
  - In each cycle, the agent retrieves the attributes of the perceived objects into the perceptual buffer and matches them against the conceptual definitions in the long term memory.

- Goals represent the objectives that an agent aims to satisfy.
  - The chosen skills within the LTSM are executed, thereby altering the environment to bring the agent closer to its goals.
  - ICARUS focuses on only one goal at a time so that only the highest priority goal not yet satisfied is attended to by the ICARUS agent.

- Whenever the agent is unable to find an applicable skill to achieve its current goal upon reaching an impasse, it employs means-ends analysis as its problem solving strategy.
4. BDI

- BDI, originally developed to reason and plan in a dynamic environment, meets real-time constraints by reducing the time used in planning and reasoning.
- A BDI agent can react to changes and communicate in their embedded environment, as it attempts to achieve its goals.
4. BDI

4.1 BDI Architecture

- Beliefs are facts about the world as well as inference rules that may lead to acquisition of new beliefs.

- Plans refer to sequences of actions that a BDI agent can perform to achieve one or more of its intentions.
  - The body of a plan comprises of possible courses of actions and procedures to achieve a goal.
  - The invocation condition specifies the prerequisites to be met for the plan to be executed, or to continue executing.

- Desires (goals) are the objectives that a BDI agent aims to accomplish.
  - A goal is said to be successfully achieved when a behavior satisfying the goal description is executed.

- Intentions are the actions that the agent is committed to perform in order to achieve the desires.
  - Each intention is implemented as a stack of plan instances, and is only considered as executable if the context of the plan matches with the consequence of the beliefs.
4.2 Functions and Processes

- A system interpreter manipulates the components of the BDI architecture
  - In each cycle, it updates the event queue by the perceptual input and the internal actions to reflect the events observed, followed by the selection of an event.
  - New possible desires are then generated by finding relevant plans in the plan library for the selected event.
  - From the set of relevant plans, an executable plan is then selected and an instance plan is thus created.
  - The instance plan is pushed onto the existing intention stack or a new intention stack.

- The BDI agent interacts with its environment either through its database when new beliefs are acquired or through actions performed during the execution of the intention.
Once the BDI architecture is committed to a process of achieving goals, it does not consider other pathways although they may be better than the one chosen. In this way, the architecture is able to cut down its decision time.

- An active goal in the BDI architecture is dropped once the system recognizes the goal as accomplished or cannot be readily accomplished.

There have been recent attempts to include learning mechanisms into BDI systems.

- Subagdja and Sonenberg (2005) further extended the BDI architecture to incorporate learning by generating and testing hypothesis for the purpose of formulating plans.
5. Subsumption Architecture

- The subsumption architecture was proposed as an incremental and bottom-up approach to deal with the problems of extensibility, robustness, and achieving multiple goals.

- The subsumption architecture *decomposes a problem in terms of the behaviors* exhibited by the robots instead of the stages of information flowing within the controller as in a traditional AI design.
5. Subsumption Architecture

5.1 Subsumption Architecture

- Subsumption allows all layers to access the sensor’s data and multiple (simple) behaviors to operate in parallel.
- Each layer is capable of controlling the system by itself, unlike classical AI agents.
- Each level of competence displays a behavior to pursue a particular goal and a higher level layer tends to subsume the underlying layers.
  - The lower layers work like fast-adapting mechanisms, allowing the agent to react quickly to changes in its environment.
  - The higher layers control the system towards the overall goals.
- The layered design is thus analogous to a biological nervous system, wherein new sections of brain are developed for new functions, but old sections are still preserved to perform their original tasks.
- The layered-based design also allows for easier implementation.
5.2 Functions and Processes

- There is no explicit representation of knowledge and therefore there is no matching of rules.
- The layers are driven by the data collected, with no global data or dynamic communication.
  - The system is being reactive via implementing an activity as a consequence of events.
  - The perception is tightly coupled to the action within each layer.
- The layers are also expected to react quickly in order to sense the rapid changes in the environment.
- However, the higher layers can suppress the inputs and inhibit the outputs of the lower layers, leading to an adjustment in the behavior for the purpose of fulfilling the overall goal.
6. CLARION

- A hybrid architecture that incorporates both implicit and explicit memories for reasoning and learning.
- Procedural knowledge can be gradually accumulated with repeated practice, and subsequently applied to practiced situations.
- It also unifies neural, reinforcement and symbolic methods to perform on-line, bottom-up learning.
6. CLARION

6.1 CLARION Architecture

- Two levels: a top level and a bottom level (reactive level)
  - The top level comprises of explicit symbolic mechanisms and the bottom level uses subsymbolic neural mechanisms.
  - *Procedural knowledge* can be acquired through RL (reinforcement learning) in a gradual and cumulative fashion, while *declarative knowledge* is acquired through rule extraction by trials and errors.

- **CLARION** architecture includes NACS (non-action-centered subsystem) and ACS (action-centered subsystem).
  - NACS contains mostly declarative knowledge, whereas ACS contains mainly procedural knowledge.
  - The top level of NACS is a general knowledge store and contains explicit representation.
  - In ACS, explicit action rules are stored in the top level.
  - The bottom level of ACS comprises of implicit decision networks, which can be trained by RL.
6. CLARION

6.2 Functions and Processes

- Reasoning takes place through comparing a known chunk with another chunk.
  - When the similarity between two chunks is sufficiently high, an inference regarding the relations between them can be made.
  - While the comparison of chunks gives rise to similarity-based reasoning, the usage of chunks during reasoning constitutes to rule-based reasoning.

- Learning can be differentiated between implicit learning (IL) or explicit learning (EL).
  - IL can be considered as learning procedural skills, and EL takes place by learning those rules as declarative knowledge.
  - The learning processes are through neural mechanisms, such as multi-layer NNs and backpropagation algorithm
  - Learning of rules in the top level takes place by extracting knowledge from the bottom level.
Functional Comparison: 1. Perception

- Soar: A Soar agent stores the information from the external environment directly into its working memory.

- ACT-R: The perceptual information from the environment enters the visual module and is made available to the central production system.

- ICARUS: It perceives objects in its environment through its perception buffer.

- BDI: Perception is available in the form of events stored in the event queue.

- Subsumption architecture: It perceives the environment through its sensors.

- CLARION: The input is a series of dimension/value pairs describing the state of the world.
## 1. Perception

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Means of perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>Perceptual input stored directly as part of the working memory</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Perception stored in visual module and made available through visual buffer</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Perceptual in perceptual buffer as part of conceptual memory</td>
</tr>
<tr>
<td>BDI</td>
<td>Perception mapped to events stored in event queue</td>
</tr>
<tr>
<td>Subsumption</td>
<td>Perception is available through sensors</td>
</tr>
<tr>
<td>CLARION</td>
<td>Perceptual input represented as dimension/value pairs</td>
</tr>
</tbody>
</table>
2. Memory

- Soar
  - Short term memory contains information received from the environment and all rules relevant to the current situation.
  - Long term memory can be procedural, declarative (semantic) or episodic and is realized through the production rule system.

- ACT-R
  - ACT-R uses a distributed memory system, wherein the goals, beliefs, sensory, and motor signals are situated in distinct buffers.
  - The production rules are stored in the procedural memory.
  - The chunks in ACT-R are knowledge stored in the declarative memory.

- ICARUS
  - The short term memory in ICARUS takes the form of belief memory updated through a conceptual inference process as well as short term skill memory.
  - Although ICARUS does not refer to declarative and procedural memories explicitly, its conceptual memory can be considered as declarative memory and the skill memory as procedural memory.
2. Memory

- **BDI**
  - The beliefs and facts about the world are stored in the database as symbolic representation of the world perceived.
  - The declarative knowledge is made available in the plan library as plan rules.
  - The procedural knowledge defines the actions to be undertaken when the plan is carried out.

- **CLARION**
  - It contains working memory as a temporary information storage to facilitate decision making.
  - The long term memory representation in CLARION differs from those of Soar and ACT-R, wherein the rules are stored as procedural knowledge.

- Subsumption architecture does not make use of neither short term memory and long term memory.
## 2. Memory

### Table 2  Implementation of memory functions in the six cognitive architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Representation</th>
<th>Working memory</th>
<th>Long term memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>Symbolic</td>
<td>Contains perceptual input, states, and production rules relevant to current situation</td>
<td>Contains procedural, declarative (semantics), and episodic memory</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Symbolic</td>
<td>Contains goal, perception, relevant knowledge, and motor action in the various buffers</td>
<td>Contains declarative knowledge in declarative module and procedural knowledge in production system</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Symbolic</td>
<td>Consists of perceptual buffer, belief memory, and short term skill memory</td>
<td>Procedural knowledge stored in long term skill memory, declarative knowledge in long term conceptual memory</td>
</tr>
<tr>
<td>BDI Subsumption</td>
<td>Symbolic</td>
<td>Belief as working memory</td>
<td>Plans as long term memory</td>
</tr>
<tr>
<td>CLARION</td>
<td>Symbolic + subsymbolic</td>
<td>As temporary information storage</td>
<td>Procedural at bottom level                                    Declarative at top level</td>
</tr>
</tbody>
</table>
3. Goals

- **Soar**
  - The goals can be divided into subgoals whenever an impasse is encountered, leading to the creation of the goal-subgoal hierarchy.

- **ACT-R**
  - Goals are stored in the intentional module and are available to the central production system via the goal buffer.
  - A goal can be decomposed into subgoals and the new goals are added into the goal stack in ACT-R.

- **ICARUS**
  - ICARUS takes a simple approach of maintaining goals in the goal memory.

- **BDI**
  - BDI supports a goal life cycle, maintaining the distinction among active goals, inactive goals, accomplished goals, abandoned goals, and unattainable goals.
3. Goals

- **Subsumption architecture**
  - Each layer works to achieve its goal(s) independently, although the higher layers can subsume the goals of lower layers.

- **CLARION**
  - A motivational subsystem creates and stores goals using a goal structure, which can be a goal stack or a goal list.
  - The goals in a goal list can be accessed randomly and they compete with each other to be the current goal.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Goal representation in the six cognitive architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Goal representation</td>
</tr>
<tr>
<td>Soar</td>
<td>Goals represented as states in working memory Support goal-subgoal hierarchy</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Goals stored in the intentional module and made available through the goal buffer</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Goals placed in the goal stack Support goal-subgoal hierarchy</td>
</tr>
<tr>
<td>BDI</td>
<td>Non-conflicting goals as desires Support goal-subgoal hierarchy</td>
</tr>
<tr>
<td>Subsumption</td>
<td>Multiple goals across modules in different layers Allow multiple and parallel goal processing</td>
</tr>
<tr>
<td>CLARION</td>
<td>Goals stored in a goal structure, such as goal stack or goal list</td>
</tr>
</tbody>
</table>
4. Problem Solving

- Soar
  - Soar implements the decision procedure to select the appropriate operator using preferences created during the elaboration phase.

- ACT-R
  - Problem solving in ACT-R occurs via the activation of chunks in the declarative memory and the retrieval of knowledge from the procedural memory.

- ICARUS
  - ICARUS places goals in the goal stack and searches for applicable skills using a backtracking strategy.

- BDI
  - The BDI architecture combines means-ends analysis with decision making to select the best plan to implement.

- Subsumption architecture
  - The actions executed by the agent are reflexive in nature, rather than being chosen via a problem solving mechanism.
4. Problem Solving

- CLARION
  - CLARION combines recommendations from the top and bottom levels to decide on appropriate reactions.
  - The process of searching for appropriate actions may be considered as a form of problem solving.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Mechanism for problem solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>Decision procedure for selecting appropriate operators; Means-ends analysis</td>
</tr>
<tr>
<td>ACT-R</td>
<td>By activation of chunks in Bayesian framework and production rule firing when chunks match with rules</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Means-ends analysis by searching and backtracking</td>
</tr>
<tr>
<td>BDI</td>
<td>Means-ends analysis</td>
</tr>
<tr>
<td>Subsumption</td>
<td>No explicit problem solving mechanism</td>
</tr>
<tr>
<td>CLARION</td>
<td>Combination of $Q$-values calculated in the bottom level and rules in the top level to choose the course of actions</td>
</tr>
</tbody>
</table>
5. Planning

- Soar
  - The decision cycle in Soar can be seen as a planning mechanism, whereby the actions are selected based on preferences to reach a goal.

- ACT-R
  - It is mentioned that subgoals are created leading to a sequence of actions to perform in a plan during the problem solving process.

- ICARUS
  - Planning is explicitly employed in ICARUS through the instantiation of skills, which is the top-down selection of skills applicable to the current beliefs for execution.

- BDI
  - Plans are stored in the plan library and intentions are plans committed by the agent for execution.
  - Each plan comprises of possible courses of action and information on situations for initiating and/or continuing its execution.
5. Planning

- **Subsumption architecture**
  - Planning within the architecture is implicit by decomposing a given task into multiple processes across layers, each can be activated with the appropriate sensory inputs.

- **CLARION**
  - Q-values are used in a method known as beam search for selecting a sequence of actions during the formulation of a plan.
  - The actions are selected such that the probability of achieving a given goal is the highest.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Mechanism for planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>Decision cycle selects appropriate actions, bringing system closer to goal</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Planning by creating subgoals</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Planning through instantiation of skills</td>
</tr>
<tr>
<td>BDI</td>
<td>Predefined plans stored in the plan library</td>
</tr>
<tr>
<td>Subsumption</td>
<td>Implicit planning by task decomposition</td>
</tr>
<tr>
<td>CLARION</td>
<td>Planning by beam search in the Q-value space</td>
</tr>
</tbody>
</table>
6. Reasoning / Inference

- **Soar**
  - When a production rule matches with the current state, the rule execution leads to an action
  - During the elaboration phase, the relevant knowledge is brought into the working memory and preferences are created, giving suggestions to which operators to choose.

- **ACT-R**
  - It uses probabilistic reasoning in the matching of production rules, after activating the relevant chunks in the declarative knowledge.

- **ICARUS**
  - It uses conceptual inference, which occurs in a bottom up, data driven manner.
  - Concepts in ICARUS are represented as Boolean structure, which are matched in an all or none manner.

- **BDI**
  - It makes use of the procedural reasoning system (PRS), which is used to select the appropriate plans for goal satisfaction.
6. Reasoning / Inference

- **Subsumption architecture**
  - A reasoning system is absent in the subsumption architecture.

- **CLARION**
  - It combines both similarity-based and rule-based mechanisms to mimic the process of human reasoning.
  - This works by comparing the similarities between chunks as part of the inference process.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Mechanism for reasoning/inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>Rule matching to bring relevant knowledge into working memory; Elaboration phase to create preferences</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Probabilistic reasoning</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Boolean match of conceptual clauses, bottom-up and data driven</td>
</tr>
<tr>
<td>BDI</td>
<td>Procedural reasoning system</td>
</tr>
<tr>
<td>Subsumption</td>
<td>Absence of reasoning mechanism</td>
</tr>
<tr>
<td>CLARION</td>
<td>Integrate rule-based reasoning and similarity-based reasoning</td>
</tr>
</tbody>
</table>
7. Learning

- **Soar**
  - When an impasse is encountered, a subgoal is created. Upon the successful resolution of the impasse, a chunk is created and added into the production memory as new knowledge.

- **ACT-R**
  - Learning takes place in the form of production compilation in ACT-R.
  - It involves a combination of various production rules into one, eventually leading to performance without the retrieval of instructions from the declarative memory

- **ICARUS**
  - Learning occurs when an impasse is reached. This allows the formation of new skills/knowledge that can be used for similar situations in the future.

- **BDI**
  - Many learning methods, such as Q-learning and learning from interpretations, have been shown to work with BDI systems.
7. Learning

- Subsumption architecture does not have learning mechanism.
- CLARION
  - Learning of procedural knowledge in the bottom level of CLARION is by multi-layer neural networks and Q-learning.
  - Learning of rules in the top level occurs through extracting rules from the bottom level.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Mechanisms for learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>Chunking for creation of new production rules</td>
</tr>
<tr>
<td></td>
<td>Reinforcement learning for updating reward of each rule</td>
</tr>
<tr>
<td></td>
<td>Episodic and semantic memory to aid in decision making</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Production compilation for combining multiple production rules into one</td>
</tr>
<tr>
<td>ICARUS</td>
<td>Skill learning</td>
</tr>
<tr>
<td>BDI</td>
<td>Q-learning, top-down induction of decision tree</td>
</tr>
<tr>
<td></td>
<td>Learning from interpretations</td>
</tr>
<tr>
<td>Subsumption</td>
<td>Absence of learning mechanism</td>
</tr>
<tr>
<td></td>
<td>Reactions to environment pre-wired into each module</td>
</tr>
<tr>
<td>CLARION</td>
<td>Q-learning at bottom level (procedural knowledge) and rule extraction at top level (declarative knowledge)</td>
</tr>
</tbody>
</table>
8. Relevance to Neurobiology

- Soar, ICARUS, BDI, and CLARION
  - They do not make explicit references to neural anatomy.

- ACT-R
  - It makes extensive references of its components to specific regions in human brains.
  - The central production system and the memory modules in ACT-R are mapped to the basal ganglia and the various cortical areas respectively.

- Subsumption architecture
  - It does not claim any significant biological connection, although its layered design is said to be consistent with that of a human nervous system.
  - The addition of new layers is similar to the development of new brain sections for new functions, leaving existing parts unaltered.
8. Relevance to Neurobiology

Table 8  The relevance of the six cognitive architectures to neurobiology

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Relevance to neurobiology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soar</td>
<td>No reference to brain anatomy</td>
</tr>
<tr>
<td>ACT-R</td>
<td>Used for prediction of brain activation pattern. Modules and production system are mapped to various brain regions</td>
</tr>
<tr>
<td>ICARUS</td>
<td>No reference to brain anatomy</td>
</tr>
<tr>
<td>BDI</td>
<td>No reference to brain anatomy</td>
</tr>
<tr>
<td>Subsumption</td>
<td>No reference to brain anatomy. Layered design is analogous to biological nervous systems</td>
</tr>
<tr>
<td>CLARION</td>
<td>Based on neural networks but no reference to brain anatomy</td>
</tr>
</tbody>
</table>
Benchmarks and Applications

- Soar has been used by US Army, Navy, and Air Force to develop robots and software for the purposes of modelling, simulations, and control.
- ACT-R has been used as a framework for various cognitive tasks, such as the Tower of Hanoi and memory for text.
- ICARUS has been applied to many cognitive tasks, including multi-column subtraction and pole balancing.
- BDI has also been applied to the problems of factory process control and business process management.
- The subsumption architecture was applied to Genghis, a six-legged walking robot, and ToTo that navigates as if it has a built-in map.
- CLARION has been used in both the simulation of navigation and cognitive tasks such as process control tasks.
## Table 9: Benchmarks and Applications

<table>
<thead>
<tr>
<th></th>
<th>Soar</th>
<th>ACT-R</th>
<th>ICARUS</th>
<th>BDI</th>
<th>Subsumption</th>
<th>CLARION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Puzzles and cognitive tasks</strong></td>
<td>Tower of Hanoi, Eight Puzzle, Fifteen Puzzle, Think-a-dot, Rubik Cube</td>
<td>Tower of Hanoi, Memory of text, Language comprehension, Communication</td>
<td>Tower of Hanoi, Peg Solitaire, Multi-column, subtraction</td>
<td>Tower of Hanoi</td>
<td>Tower of Hanoi, Serial reaction time task, Grammar learning, Alpha-betical arithmetic</td>
<td></td>
</tr>
<tr>
<td><strong>ECA, training and diagnosis</strong></td>
<td>Operation and maintenance of engines, leadership training</td>
<td></td>
<td></td>
<td></td>
<td>Museum guide, diagnosis of space shuttles</td>
<td></td>
</tr>
<tr>
<td><strong>Command control and virtual games</strong></td>
<td>TacAir-Soar, RWA-Soar, MOUTBOT for US military, Eater's World</td>
<td>Aircraft control</td>
<td>In-City driving, Pole balancing</td>
<td>Factory process control, Business process management</td>
<td>Truckin' Game, airplane control</td>
<td>Maze and minefield navigation, process control</td>
</tr>
<tr>
<td><strong>Robotics</strong></td>
<td>Robotics for US military</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


The ultimate goal of studying integrated cognitive architectures is to build coherent systems that display a wide range of functions robustly across different problem settings.

Though the existing architectures have been increasingly applicable to many tasks, there remains a challenge to build one system capable of solving all types of problems in a real environment.

It would be better if the architectures were applicable to universal situations so that the agent is able to solve problems in all kinds of novel situations, and not just confined to test situations.
1. Converging Design Principles

- All cognitive architectures bear important similarities in their principles and approaches.
  - The use of STWM is critical to hold the relevant input from the environment and the knowledge from the LTM for the selection of appropriate actions.
  - Most cognitive architectures make use of some forms of rules to aid in selecting the appropriate courses of action.
  - The decision processes in Soar, ACT-R and ICARUS are also similar, which involve the selection of skills/actions based on the desired goals and the information available in the STM.

STWM: short term working memory
LTM: long term memory
Most of the architectures make a distinction between declarative and procedural knowledge.

- Declarative knowledge contains the facts and inference rules required for reasoning.
- Procedural knowledge encodes the sequences of action to perform in response to specific situations.
2. Promising Research Trends

- Real-time embodiment
  - Some of the architectures are designed to remain responsive even in a dynamic environment, therefore enabling situatedness.
  - These architectures are able to sense changes in the environment, as well as respond rapidly to the changes.
  - We are eager to witness the development of truly embodied intelligent systems, that are capable of sensing and interacting with the environment in real time, just like human beings.
Interaction based learning

- A cognitive agent should be able to learn and function in a real-time dynamic environment.

- In recent years, a family of self-organizing neural models, known as fusion ART and TD-FALCON (Temporal Difference-Fusion Architecture for Learning and Cognition) have been steadily developed.

- Such models may provide a promising approach to designing cognitive systems for functioning and learning in real-time environment.
Evaluation

- There is a lack of standard benchmarks and problem domains for comparing the various systems side by side at the system level.
- It is essential to have a collection of general domain tasks, each demanding a variety of cognitive functionalities, based on which the various architecture can be evaluated.

Biologically-inspired architectures

- One key objective of developing integrated cognitive architectures is to mimic and explain human cognition.
- While powerful constructs, like problem solving and goal-subgoal hierarchy, have been available for a long time, little is known how to achieve the same level of capabilities in a biological based architecture.
- The answer to this question could well be the key to the next generation integrated cognitive architectures.

Conclusion